GPU Implementation of a Krylov solver
Preconditioned by a Shifted Laplace Multigrid
Method for Helmholtz Equation

Hans Knibbe*, Kees Vuik*, Kees Oosterlee#

* Delft University of Technology
# Dutch national research centre for mathematics and computer science (CWI)
Outline

1. Motivation
2. Problem Description
3. Solver
4. GPU implementation
   - Single GPU
   - Multi-GPU
5. Results
Geophysical Background

Time Domain

Frequency Domain

FFT

Frequency 1

Frequency 2

...  

Frequency n

IFFT
Marmousi 2D
Motivation
Problem Description

The Helmholtz equation in three dimensions is considered

\[- \frac{\partial^2 \phi}{\partial x^2} - \frac{\partial^2 \phi}{\partial y^2} - \frac{\partial^2 \phi}{\partial z^2} - (1 - \alpha i) k^2 \phi = g,\]  

(1)

with non-reflecting first-order radiation boundary conditions

\[ \left( - \frac{\partial}{\partial \eta} - ik \right) \phi = 0, \]

(2)

where \( \phi = \phi(x, y, z) \) is the wave pressure field, \( k = k(x, y, z) \) is the wavenumber, \( \alpha \) is the damping coefficient, \( g = g(x, y, z) \) is the source term.
Bi-CGSTAB preconditioned by shifted Laplace multigrid method. Equation solved in preconditioner is

\[-\frac{\partial^2 \phi}{\partial x^2} - \frac{\partial^2 \phi}{\partial y^2} - \frac{\partial^2 \phi}{\partial z^2} - (\beta_1 - \beta_2 i)k^2 \phi = g, \quad (3)\]

\[\beta_1, \beta_2 \in \mathbb{R}, \text{ with the same boundary conditions as the original problem.}\]

Multigrid components:
- Standard restriction
- Multi-coloured Gauss-Seidel as a smoother

Preconditioner is computed in single precision.
1. Motivation

2. Problem Description

3. Solver

4. GPU implementation
   - Single GPU
   - Multi-GPU

5. Results
Little-Green Machine

20 general computing nodes
- 2 Intel quadcore E5620
- 24 GB RAM
- 2 TB disk
- 2 NVIDIA GTX480

Founded by
- University of Leiden
- NWO
- TU Delft
- KNMI
**Sparse Matrix-Vector Multiplication**

\[ x_i = \sum_{j=1}^{N} A_{ij} b_j \]

<table>
<thead>
<tr>
<th>Size</th>
<th>Time 8-cores (ms)</th>
<th>Time GPU (ms)</th>
<th>CPU/GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>1.5</td>
<td>0.1</td>
<td>15</td>
</tr>
<tr>
<td>100,000</td>
<td>8.4</td>
<td>0.1</td>
<td>84</td>
</tr>
<tr>
<td>1,000,000</td>
<td>64.9</td>
<td>1.1</td>
<td>59</td>
</tr>
<tr>
<td>5,000,000</td>
<td>328.1</td>
<td>5.4</td>
<td>61</td>
</tr>
<tr>
<td>20,000,000</td>
<td>1334.6</td>
<td>22</td>
<td>61</td>
</tr>
</tbody>
</table>
Gauss-Seidel Smoother

For each color compute

\[ x_i = (1 - \omega)x_i + \omega(b_i - \frac{\sum_j A_{ij}x_j}{A_{ii}}), \]

<table>
<thead>
<tr>
<th>Size</th>
<th>Time 8-cores (ms)</th>
<th>Time GPU (ms)</th>
<th>CPU/GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,000</td>
<td>4</td>
<td>0.6</td>
<td>7</td>
</tr>
<tr>
<td>100,000</td>
<td>23.4</td>
<td>0.8</td>
<td>29</td>
</tr>
<tr>
<td>1,000,000</td>
<td>164.5</td>
<td>2.5</td>
<td>66</td>
</tr>
<tr>
<td>5,000,000</td>
<td>625.9</td>
<td>10.3</td>
<td>61</td>
</tr>
<tr>
<td>20,000,000</td>
<td>3733.9</td>
<td>39.4</td>
<td>95</td>
</tr>
</tbody>
</table>
1 Motivation

2 Problem Description

3 Solver

4 GPU implementation
   - Single GPU
   - Multi-GPU

5 Results
Multi-GPU

Pushing context

Multi-threading

CPU

Context

GPU GPU GPU GPU

CPU

Threads

GPU GPU GPU GPU
NVidia Computer

8 GPUs, each GPU has 448 cores, 3 GB RAM
12 cores (2 Westmeres), 48 GB RAM
Multi-GPU Approach

1. **Data-parallel approach** (e.g. vector operations on multi-GPU)
   - Relatively easy to implement
   - CPU → GPU → CPU data transfer

2. **Split of the algorithm** (e.g. solver on one GPU, preconditioner on the another one)
   - No or little data transfers
   - Find the best way to split the algorithm

3. **Domain-Decomposition approach** (e.g. each domain on a different GPU)
   - Exchange of halos (still data transfer)
   - Can affect convergence of the preconditioned method
Multigrid on Multi-GPU
Multi-GPU Issues

- Limited GPU memory size so need multiple GPUs for large problems.
- Efficient memory reusage to avoid allocation/deallocation, e.g. pool of GPU-vectors.
- Limit communications CPU → GPU and GPU → CPU.
- Each GPU need separate texture reference.
- Cublas vectors limited to 512 MB.
Speedups for Sparse MVM

Row-wise splitting

Speedups single GPU versus Multi-GPU

<table>
<thead>
<tr>
<th>Size</th>
<th>12-cores/1 GPU</th>
<th>12-cores/8-GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000,000</td>
<td>51.9</td>
<td>10.3</td>
</tr>
<tr>
<td>30,000,000</td>
<td>47.4</td>
<td>10.4</td>
</tr>
<tr>
<td>100,000,000</td>
<td>-</td>
<td>6.8</td>
</tr>
</tbody>
</table>
Gauss-Seidel Smoother

For each color

\[ x_i = (1 - \omega)x_i + \omega (b_i - \frac{\sum_j A_{ij}x_j}{A_{ii}}), \]

<table>
<thead>
<tr>
<th>Size</th>
<th>12-cores/1 GPU</th>
<th>12-cores/8-GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000,000</td>
<td>16.5</td>
<td>5.2</td>
</tr>
<tr>
<td>30,000,000</td>
<td>89.1</td>
<td>6.1</td>
</tr>
<tr>
<td>100,000,000</td>
<td>-</td>
<td>4.4</td>
</tr>
</tbody>
</table>
## Bi-CGSTAB

### Timings for Bi-CGSTAB, single precision

<table>
<thead>
<tr>
<th>n</th>
<th>12-cores</th>
<th>1 GPU</th>
<th><strong>Speedup</strong></th>
<th>8-GPU</th>
<th><strong>Speedup</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000,000</td>
<td>24 s</td>
<td>0.8 s</td>
<td><strong>29.8</strong></td>
<td>2.3 s</td>
<td><strong>10.5</strong></td>
</tr>
<tr>
<td>15,000,000</td>
<td>82 s</td>
<td>2 s</td>
<td><strong>38.1</strong></td>
<td>5.8 s</td>
<td><strong>14.2</strong></td>
</tr>
<tr>
<td>100,000,000</td>
<td>395 s</td>
<td>-</td>
<td>-</td>
<td>28.6 s</td>
<td><strong>13.8</strong></td>
</tr>
</tbody>
</table>
Wedge problem
### Bi-CGSTAB preconditioned by shifted Laplace multigrid

Problem size $350 \times 350 \times 350 \approx 43,000,000$ unknowns

<table>
<thead>
<tr>
<th></th>
<th>Bi-CGSTAB (DP)</th>
<th>Preconditioner (SP)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-cores</td>
<td>94 s</td>
<td>690 s</td>
<td>784 s</td>
</tr>
<tr>
<td>1 GPU</td>
<td>13 s</td>
<td>47 s</td>
<td>60 s</td>
</tr>
<tr>
<td>Speedup CPU/GPU</td>
<td>7.2</td>
<td>14.7</td>
<td>13.1</td>
</tr>
<tr>
<td>8 GPUs</td>
<td>83 s</td>
<td>86 s</td>
<td>169 s</td>
</tr>
<tr>
<td>Speedup CPU/GPUs</td>
<td>1.1</td>
<td>7.9</td>
<td>4.6</td>
</tr>
<tr>
<td>2 GPUs+split</td>
<td>12 s</td>
<td>38 s</td>
<td>50 s</td>
</tr>
<tr>
<td>Speedup CPU/GPU</td>
<td>7.8</td>
<td>18.2</td>
<td>15.5</td>
</tr>
</tbody>
</table>
Acknowledgments

- NVIDIA for access to the latest many-core-multi-GPU architecture
- TU Delft for access to the cluster
Thank you for your attention!
Sparse MVM on Multi-GPU

Row-wise splitting

Column-wise splitting